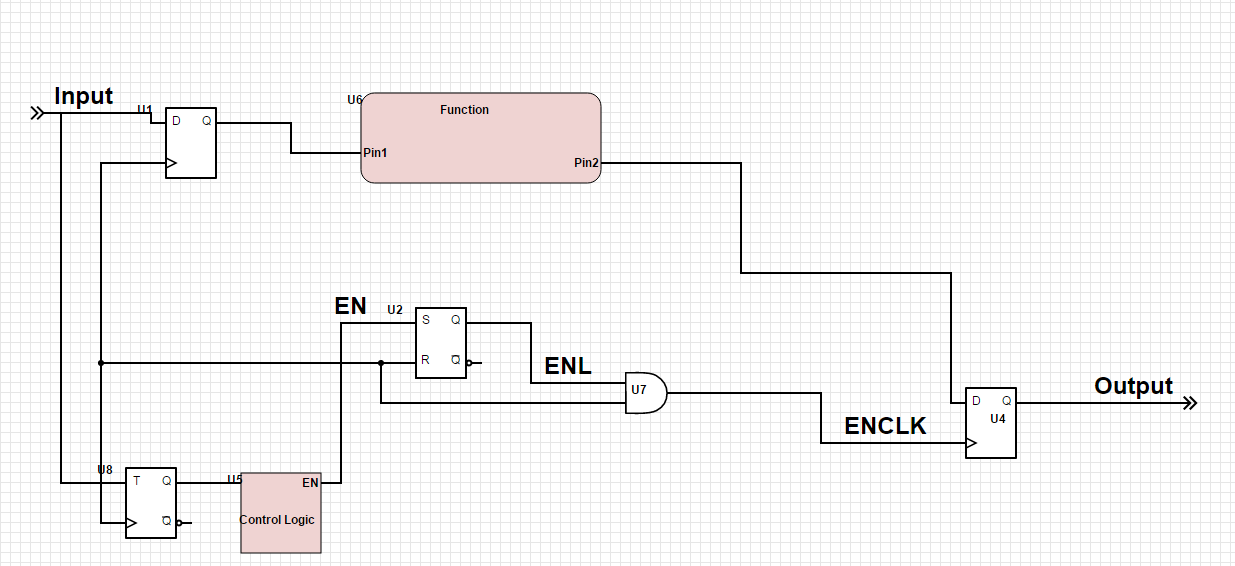
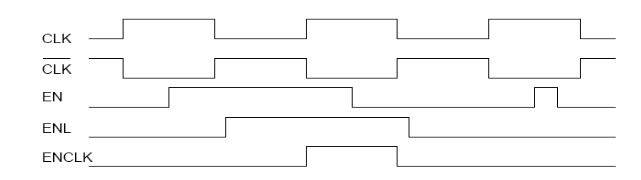
Lab 9 Report

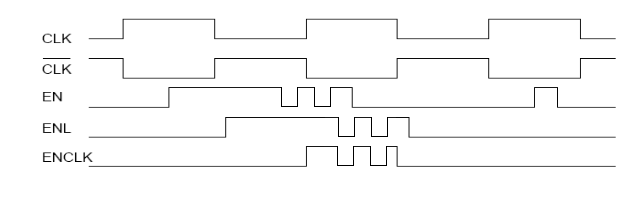
The task of this lab is to reduce the power consumption of the circuit. In order to do this, we will eliminate any unnecessary switching of registers or gates. There are many possible registers that may be clock gated, but we will only do this for the output registers for time purposes.

# 1 DFF-based output clock gating

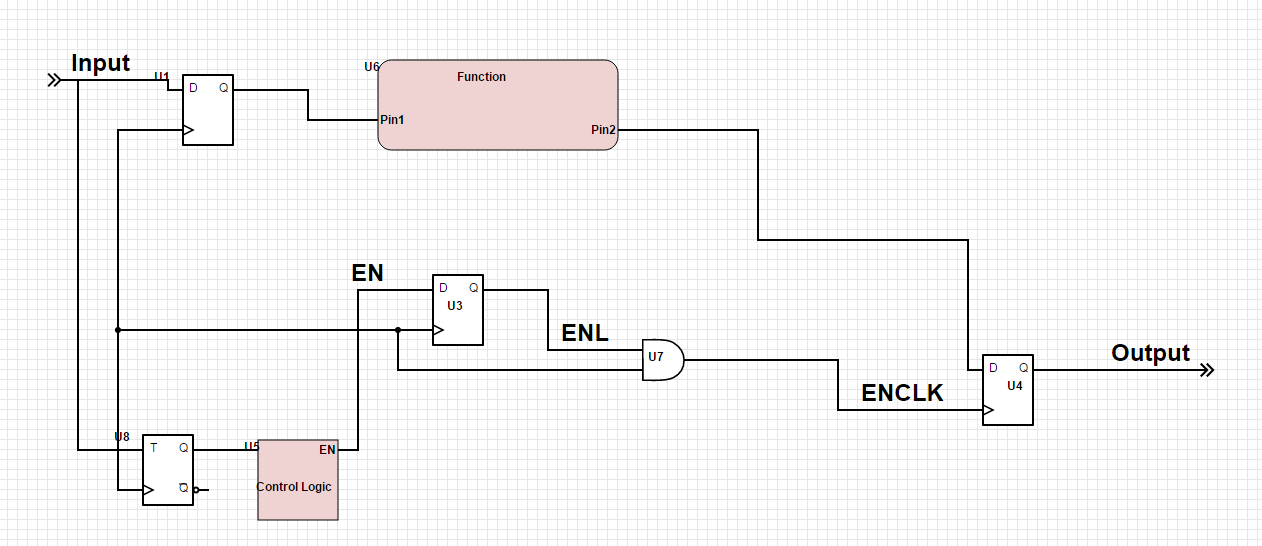
As you can see, this design uses a latch and an AND gate to clock-gate the output registers.  It will produce an output like this..

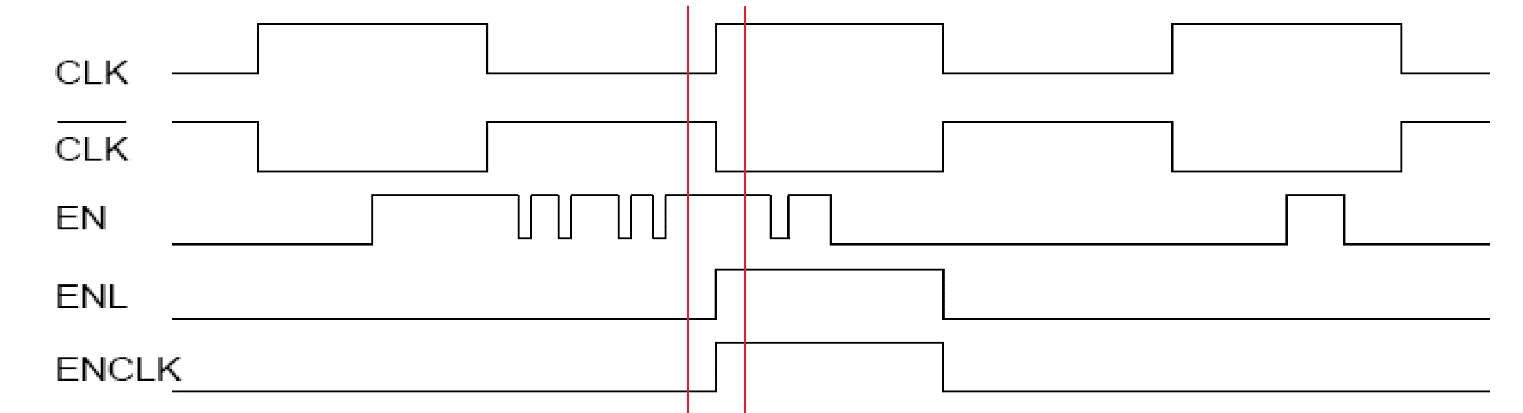


But if glitches are present in the control logic signal, there will be major problems as shown below with multiple clock edges at the output:



So.. If we use a D-flip-flop instead, there will be no glitches at the output.

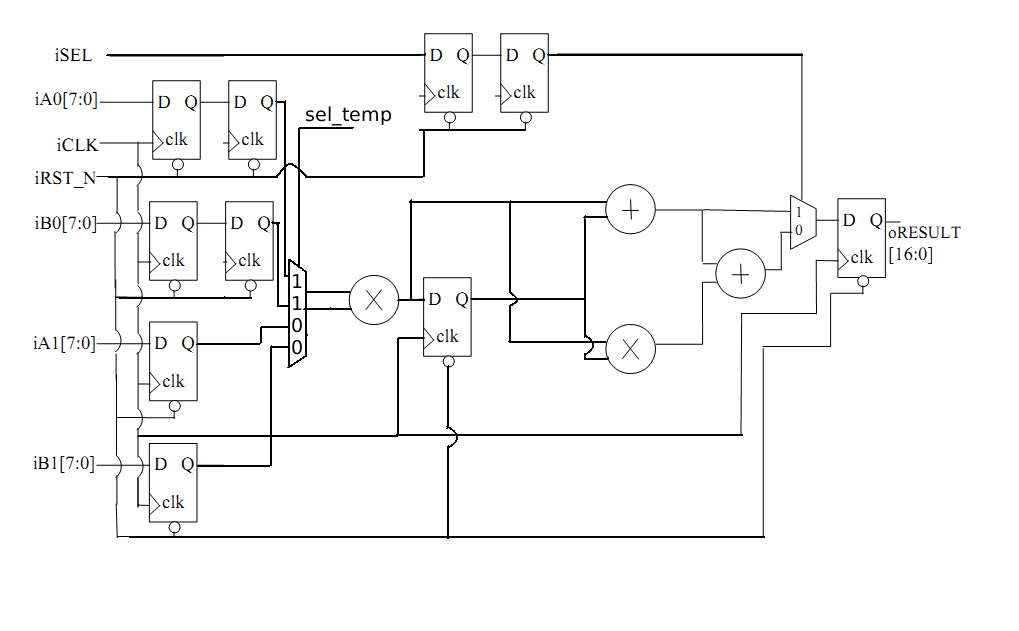


You can see that as long as the enable signal is high at the positive clock edge, the circuit will function properly. 

# 2/3 Power-Efficient Design changes.

## Before Change:

The schematic diagram for the circuit before our changes for power efficiency is below:



**module** ALT\_MULTADD\_re**(**iCLK**,** iRST\_N**,** iSEL**,** iSEL2**,** iA0**,** iA1**,** iB0**,** iB1**,** oR**);**

**input** iCLK**,** iRST\_N**,** iSEL**,** iSEL2**,** iA0**,** iA1**,** iB0**,** iB1**;**

**output** oR**;**

**wire** iCLK**,** iRST\_N**,** iSEL**,** iSEL2**;**

**wire** **[**7**:**0**]** iA0**,** iA1**,** iB0**,** iB1**;**

**reg** SEL1**,** SEL2**,** sel\_tmp**;**

**reg** **[**7**:**0**]** A0**,** A1**,** B0**,** B1**,** A01**,** B01**,** AxB**;**

**reg** **[**16**:**0**]** oR**;**

**always** **@** **(posedge** iCLK **or** **negedge** iRST\_N**)begin**

A0 **<=** iA0**;**

B0 **<=** iB0**;**

A1 **<=** iA1**;**

B1 **<=** iB1**;**

A01 **<=** A0**;**

B01 **<=** B0**;**

SEL1 **<=** iSEL**;**

SEL2 **<=** SEL1**;**

sel\_tmp **<=** iSEL2**;**

**if(**iRST\_N**)begin**

**if(~**sel\_tmp**)begin**

AxB **<=** A01 **\*** B01**;**

**end**

**else** **begin**

AxB **<=** A1 **\*** B1**;**

**end**

**if(**SEL2**)** **begin**

oR **<=** AxB **+** A1 **\*** B1**;**

**end**

**else** **begin**

oR **<=** AxB **+** A1 **\*** B1 **+** AxB **\*** A1 **\*** B1**;**

**end**

**end**

**else** **begin**

sel\_tmp **<=** 0**;**

oR **<=** 0**;**

**end**

**end**

**endmodule**

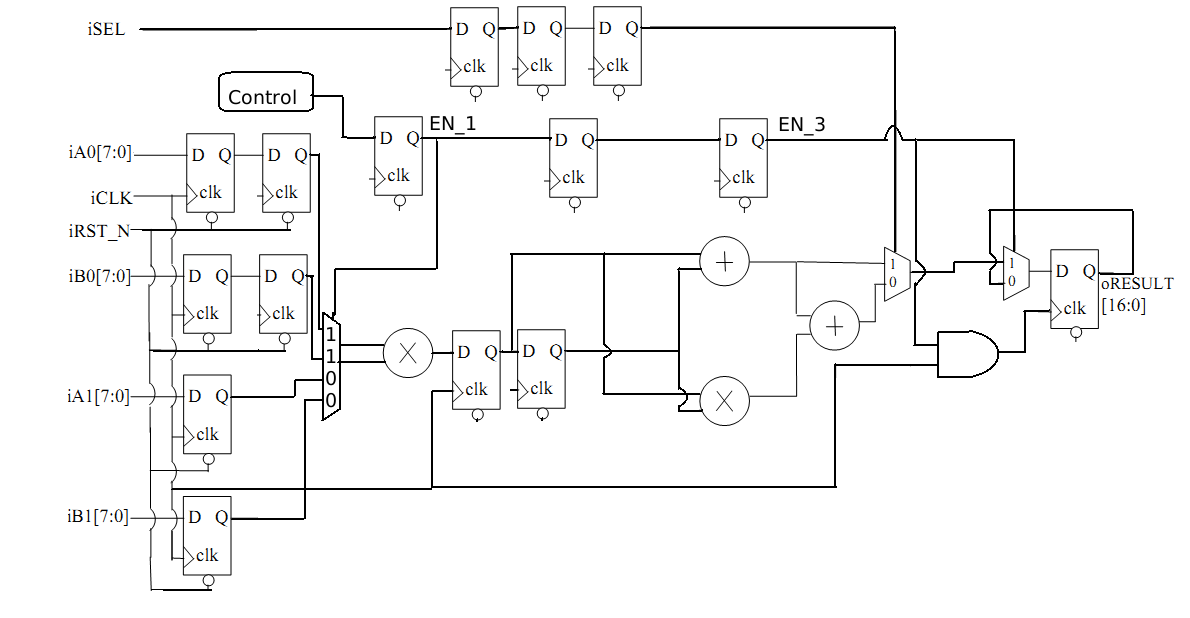
With this version of my code, there is no control logic, so the iSEL\_TMP signal serves as the indicator that a new input is present and must be a one clock cycle pulse. I intend to create control logic to create this signal automatically depending on the input signal edges.

## After Change:

I decided to re-write the Verilog code to make it easier for the compiler to know to exactly construct the circuit.

I decided to re-introduce the internal control logic that would be triggered by edges on any of the input signals iA0,iA1,iB0,iB1 and iSEL. This would make the control logic for the enable signal for the clock gate much easier.

I also added the optional d-flip flop after the first multiplier to fully pipeline the circuit.



As you can see, I changed the clock frequency to 5MHz by setting the timescale to 100ns which is half the clock cycle. 1s/200ns = 5MHz.

`timescale 100ns**/**1ns

**module** ALT\_MULTADD\_9**(**iCLK**,** iRST\_N**,** iSEL**,** iA0**,** iA1**,** iB0**,** iB1**,** oRESULT**);**

**input** iCLK**,** iRST\_N**,** iSEL**,** iA0**,** iA1**,** iB0**,** iB1**;**

**output** oRESULT**;**

**wire** iCLK**,** iRST\_N**,** iSEL**;**

**wire** **[**7**:**0**]** iA0**,** iA1**,** iB0**,** iB1**;**

**reg** SEL1**,** SEL2**,** SEL3**,** EN**,** EN\_1**,** EN\_2**,** EN\_3**;**

**reg** **[**7**:**0**]** A0**,** A1**,** B0**,** B1**,** A01**,** B01**,** A**,** B**;**

**reg** **[**16**:**0**]** oR**,** AB**,** oRESULT**,** AB1\_FF**,** AB0\_FF**;**

**reg** go**,** no**,** no\_FF**,** go\_FF**;**

**always** **@** **(posedge** iCLK**)begin**

**if(~**iRST\_N**)begin**

A0 **<=** 0**;**

B0 **<=** 0**;**

A1 **<=** 0**;**

B1 **<=** 0**;**

A01 **<=** 0**;**

B01 **<=** 0**;**

SEL1 **<=** 0**;**

SEL2 **<=** 0**;**

SEL3 **<=** 0**;**

EN **=** 0**;**

EN\_1 **<=** 0**;**

EN\_2 **<=** 0**;**

EN\_3 **<=** 0**;**

AB **=** 0**;**

AB1\_FF **<=** 0**;**

AB0\_FF **<=** 0**;**

go\_FF **<=** 0**;**

no\_FF **<=** 0**;**

go **=** 0**;**

no **=** 0**;**

oR **=** 0**;**

oRESULT **<=** 0**;**

**end**

**else** **begin**

//Input registers

A0 **<=** iA0**;**

B0 **<=** iB0**;**

A1 **<=** iA1**;**

B1 **<=** iB1**;**

A01 **<=** A0**;**

B01 **<=** B0**;**

//Multiplied signal registers

AB1\_FF **<=** AB**;**

AB0\_FF **<=** AB1\_FF**;**

//Select signal registers

SEL1 **<=** iSEL**;**

SEL2 **<=** SEL1**;**

SEL3 **<=** SEL2**;**

//enable(multiplexer select) signal registers

EN\_1 **<=** EN**;**

EN\_2 **<=** EN\_1**;**

EN\_3 **<=** EN\_2**;**

//control logic stuff

go\_FF **<=** go**;**

no\_FF **<=** no**;**

**if(**EN\_3**)** **begin**

oRESULT **<=** oR**;** //clock gate

**end**

//output register

**end**

**end**

//Select Control Logic

**always@** **(**go\_FF **or** EN\_1 **or** no\_FF**)** **begin** //

**if(**go\_FF **^** no\_FF**)** **begin** //whenever no\_FF and go\_FF are different, that means there is a new input.

EN **=** 1**;**

no **=** **~**no\_FF**;**

**end**

**if(**EN\_1**)begin**

EN **=** 0**;**

**end**

**end**

// part of control logic

**always** **@** **(**iA0 **or** iB0 **or** iA1 **or** iB1 **or** iSEL**)** **begin**

go **=** **~**go\_FF**;**

**end**

//first stage of logic (multiplier)

**always@** **(**EN\_2 **or** A01 **or** B01 **or** A1 **or** B1**)** **begin** //

**if(**EN\_2**)begin** // this is the “multiplexer” for the multiplier

A **=** A01**;**

B **=** B01**;**

**end**

**else** **begin**

A **=** A1**;**

B **=** B1**;**

**end**

AB **=** A **\*** B**;**

**end**

// final stage of logic

**always@** **(**A01 **or** B01 **or** A1 **or** B1 **or** AB1\_FF **or** AB0\_FF**)** **begin** //

**if(**SEL3**)** **begin** // this is the “multiplexer” 1 input of interest.

oR **=** AB0\_FF **+** AB1\_FF**;**

**end**

**else** **begin** // this is the “multiplexer” 0 input of interest.

oR **=** AB0\_FF **+** AB1\_FF **+** AB0\_FF **\*** AB1\_FF**;**

**end**

**end**

**endmodule**

We implemented the control logic with an always block triggered by changes in the inputs. We made 3 one bit registers go, no and count. When go and no are different, which will happen when the input changes because of the always block that inverts go, count will be set to 1 as well as no being inverted as well.

# 4 Verify

We made this test bench which would test the throughput of the circuit and many different inputs:

After the initial input, we delayed the next input by #4 which means 2 clock cycles. We were able to get correct outputs with this so this means the expected maximum throughput is verified.

`timescale 100ns**/**1ns

**module** ALT\_MULTADD\_TB **();**

**reg** iCLK\_t**,** iRST\_N\_t**,** iSEL\_t**;**

**reg** **[**7**:**0**]** iA0\_t**,** iA1\_t**,** iB0\_t**,** iB1\_t**;**

**wire** **[**16**:**0**]** oRESULT\_t**;**

ALT\_MULTADD\_9 X**(**iCLK\_t**,** iRST\_N\_t**,** iSEL\_t**,** iA0\_t**,** iA1\_t**,** iB0\_t**,** iB1\_t**,** oRESULT\_t**);**

**initial** $display **(**"Test control"**);**

**initial** $display **(**"%10s %10s %10s %10s %10s %10s %10s %10s"**,**

"iCLK\_t"**,** "iRST\_N\_t"**,** "iSEL\_t"**,** "iA0\_t"**,** "iA1\_t"**,**

"iB0\_t"**,** "iB1\_t"**,** "oRESULT\_t"**);**

**initial** $monitor **(**"%10b %10b %10b %10d %10d %10d %10d %10d"**,**

iCLK\_t**,** iRST\_N\_t**,** iSEL\_t**,** iA0\_t**,** iA1\_t**,** iB0\_t**,** iB1\_t**,** oRESULT\_t**);**

**initial** **begin**

iCLK\_t **=** 0**;**

iRST\_N\_t **=** 0**;**

iSEL\_t **=** 0**;**

iA0\_t **=** 0**;**

iA1\_t **=** 0**;**

iB0\_t **=** 0**;**

iB1\_t **=** 0**;**

**#**7

iRST\_N\_t **=** 1**;**

**#**1

iA0\_t **=** 3**;**

iA1\_t **=** 2**;**

iB0\_t **=** 3**;**

iB1\_t **=** 2**;**

**#**4

iSEL\_t **=** 1**;**

iA0\_t **=** 1**;**

iA1\_t **=** 1**;**

iB0\_t **=** 1**;**

iB1\_t **=** 1**;**

**#**4

iA0\_t **=** 2**;**

iA1\_t **=** 2**;**

iB0\_t **=** 2**;**

iB1\_t **=** 2**;**

**#**4

iSEL\_t **=** 0**;**

iA0\_t **=** 2**;**

iA1\_t **=** 2**;**

iB0\_t **=** 2**;**

iB1\_t **=** 6**;**

**#**4

iA0\_t **=** 1**;**

iA1\_t **=** 1**;**

iB0\_t **=** 1**;**

iB1\_t **=** 1**;**

**#**4

iSEL\_t **=** 1**;**

iA0\_t **=** 2**;**

iA1\_t **=** 2**;**

iB0\_t **=** 3**;**

iB1\_t **=** 3**;**

**#**4

iSEL\_t **=** 0**;**

iA0\_t **=** 1**;**

iA1\_t **=** 1**;**

iB0\_t **=** 1**;**

iB1\_t **=** 1**;**

**#**4

iA0\_t **=** 26**;**

iA1\_t **=** 1**;**

iB0\_t **=** 15**;**

iB1\_t **=** 19**;**

**#**4

iSEL\_t **=** 1**;**

iA0\_t **=** 26**;**

iA1\_t **=** 1**;**

iB0\_t **=** 15**;**

iB1\_t **=** 19**;**

**#**4

iSEL\_t **=** 1**;**

iA0\_t **=** 1**;**

iA1\_t **=** 2**;**

iB0\_t **=** 3**;**

iB1\_t **=** 4**;**

**#**4

iSEL\_t **=** 0**;**

iA0\_t **=** 1**;**

iA1\_t **=** 2**;**

iB0\_t **=** 3**;**

iB1\_t **=** 4**;**

**#**4

iSEL\_t **=** 1**;**

iA0\_t **=** 4**;**

iA1\_t **=** 3**;**

iB0\_t **=** 2**;**

iB1\_t **=** 1**;**

**#**4

iSEL\_t **=** 0**;**

iA0\_t **=** 4**;**

iA1\_t **=** 3**;**

iB0\_t **=** 2**;**

iB1\_t **=** 1**;**

**#**20

$stop**;**

**end**

**always** **#**1 iCLK\_t **=** **~**iCLK\_t**;**

**endmodule**

**# Test control**

**# iCLK\_t iRST\_N\_t iSEL\_t iA0\_t iA1\_t iB0\_t iB1\_t oRESULT\_t**

**# 0 0 0 0 0 0 0 x**

**# 1 0 0 0 0 0 0 0**

**# 0 0 0 0 0 0 0 0**

**# 1 0 0 0 0 0 0 0**

**# 0 0 0 0 0 0 0 0**

**# 1 0 0 0 0 0 0 0**

**# 0 0 0 0 0 0 0 0**

**# 1 1 0 0 0 0 0 0**

**# 0 1 0 3 2 3 2 0**

**# 1 1 0 3 2 3 2 0**

**# 0 1 0 3 2 3 2 0**

**# 1 1 0 3 2 3 2 0**

**# 0 1 1 1 1 1 1 0**

**# 1 1 1 1 1 1 1 0**

**# 0 1 1 1 1 1 1 0**

**# 1 1 1 1 1 1 1 0**

**# 0 1 1 2 2 2 2 0**

**# 1 1 1 2 2 2 2 49**

**# 0 1 1 2 2 2 2 49**

**# 1 1 1 2 2 2 2 49**

**# 0 1 0 2 2 2 6 49**

**# 1 1 0 2 2 2 6 2**

**# 0 1 0 2 2 2 6 2**

**# 1 1 0 2 2 2 6 2**

**# 0 1 0 1 1 1 1 2**

**# 1 1 0 1 1 1 1 8**

**# 0 1 0 1 1 1 1 8**

**# 1 1 0 1 1 1 1 8**

**# 0 1 1 2 2 3 3 8**

**# 1 1 1 2 2 3 3 64**

**# 0 1 1 2 2 3 3 64**

**# 1 1 1 2 2 3 3 64**

**# 0 1 0 1 1 1 1 64**

**# 1 1 0 1 1 1 1 3**

**# 0 1 0 1 1 1 1 3**

**# 1 1 0 1 1 1 1 3**

**# 0 1 0 26 1 15 19 3**

**# 1 1 0 26 1 15 19 12**

**# 0 1 0 26 1 15 19 12**

**# 1 1 0 26 1 15 19 12**

**# 0 1 1 26 1 15 19 12**

**# 1 1 1 26 1 15 19 3**

**# 0 1 1 26 1 15 19 3**

**# 1 1 1 26 1 15 19 3**

**# 0 1 1 1 2 3 4 3**

**# 1 1 1 1 2 3 4 7819**

**# 0 1 1 1 2 3 4 7819**

**# 1 1 1 1 2 3 4 7819**

**# 0 1 0 1 2 3 4 7819**

**# 1 1 0 1 2 3 4 409**

**# 0 1 0 1 2 3 4 409**

**# 1 1 0 1 2 3 4 409**

**# 0 1 1 4 3 2 1 409**

**# 1 1 1 4 3 2 1 11**

**# 0 1 1 4 3 2 1 11**

**# 1 1 1 4 3 2 1 11**

**# 0 1 0 4 3 2 1 11**

**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**

**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**

**# 1 1 0 4 3 2 1 11**

**# 0 1 0 4 3 2 1 11**

**# 1 1 0 4 3 2 1 11**

**# 0 1 0 4 3 2 1 11**

**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**

**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**

**# 1 1 0 4 3 2 1 35**

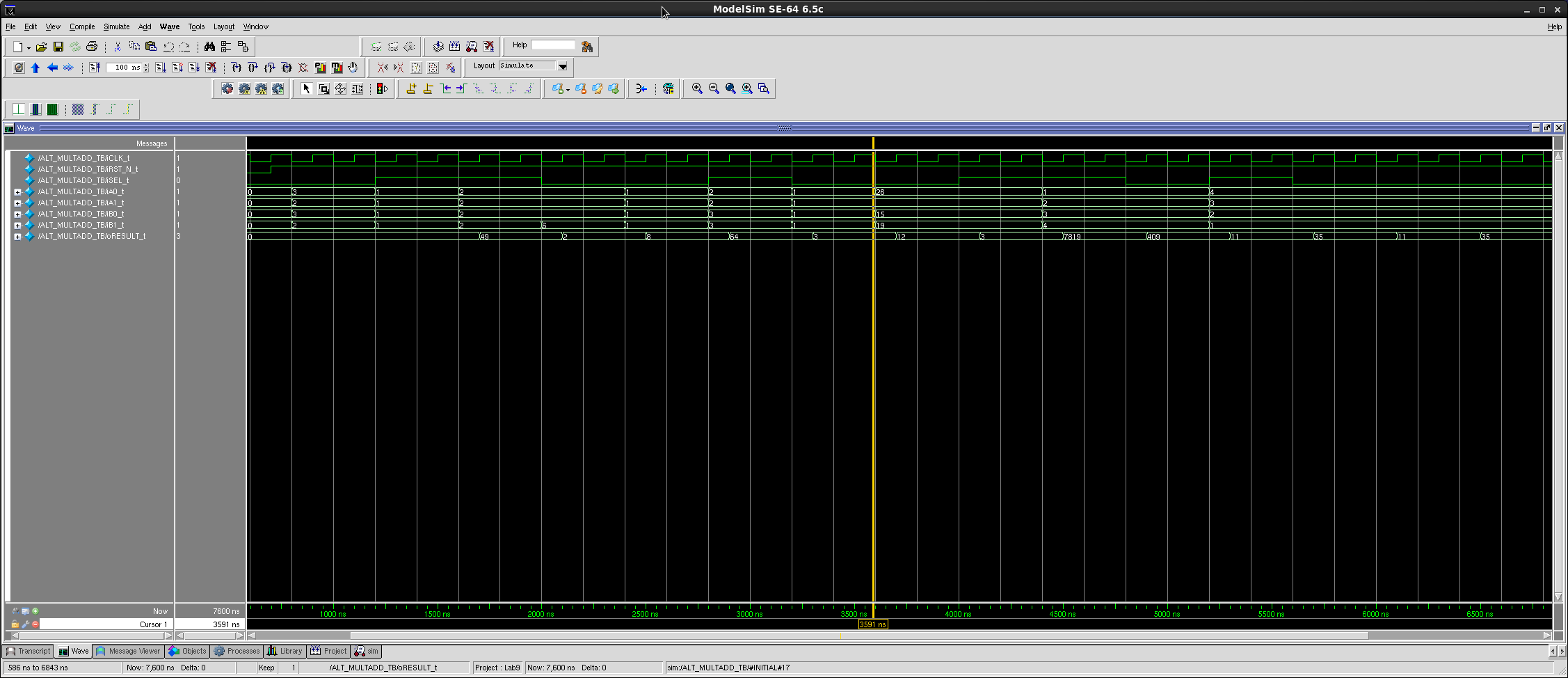
**# 0 1 0 4 3 2 1 35**

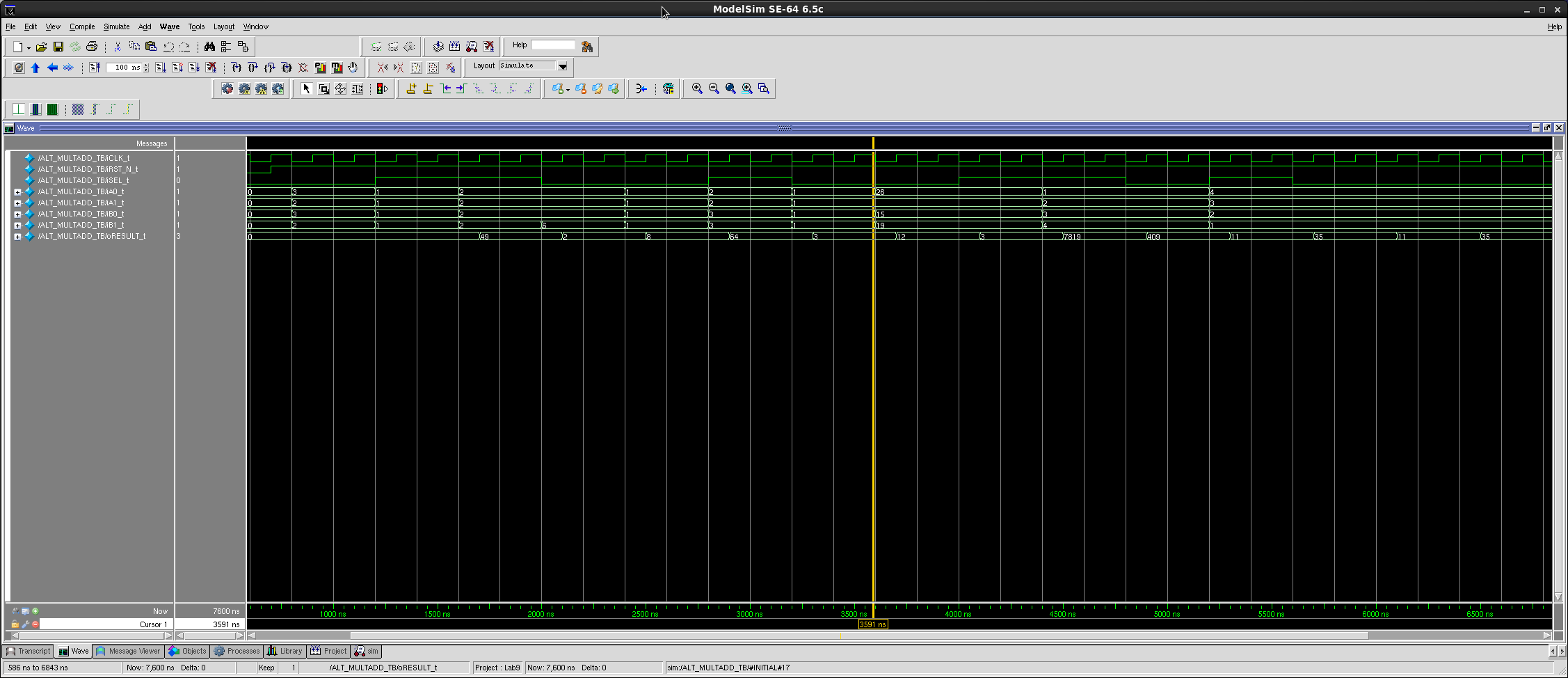
**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**

**# 1 1 0 4 3 2 1 35**

**# 0 1 0 4 3 2 1 35**





# 5 Synthesis:

In order to synthesize the Verilog code, I used this script.

## This sets the name of the directory in which area/timing/power reports

## and synthesized (mapped) netlists are stored.

**set** OUTPUT\_DIR **./**run\_dir

**if** **{** **![file** exists **${**OUTPUT\_DIR**}]** **}** **{** sh mkdir **${**OUTPUT\_DIR**}** **}**

#### Step 1 ####

## This tells the compiler where to look for the libraries

set\_attribute lib\_search\_path **../**libdir

## This defines the libraries to use

set\_attribute library **{**tcbn65gpluswc.lib**}**

##set\_attribute library {tcbn65gplustc.lib}

##set\_attribute library {tcbn90ghpbc\_ccs.lib}

##set\_attribute lp\_insert\_clock\_gating true

#set\_attribute lp\_insert\_operand\_isolation true

#rm /designs/\*

#### Step 2 ####

##This must point to your VHDL/verilog file

**load** -v2001 **../../**ALT\_MULTADD\_redid.v

set\_attribute lp\_insert\_clock\_gating false

#### Step 3 ####

## This builds the general block

elaborate

dc::set\_time\_unit -nanoseconds

dc::set\_load\_unit -picofarads

define\_clock -period 200 -name clk **[**dc::get\_ports **{**iCLK**}]** -rise 5 -fall 5

set\_attribute lp\_power\_unit **{**nW**}**

set\_attribute max\_leakage\_power 100000 **/**designs**/**ALT\_MULTADD\_9

set\_attribute power\_optimization\_effort high

synthesize -to\_mapped -effort high

report area **>** **${**OUTPUT\_DIR**}/**area.rpt

report gates **>** **${**OUTPUT\_DIR**}/**gates.rpt

report timing **>** **${**OUTPUT\_DIR**}/**timing.rpt

report timing -lint **>** **${**OUTPUT\_DIR**}/**lint.rpt

report summary **>** **${**OUTPUT\_DIR**}/**summary.rpt

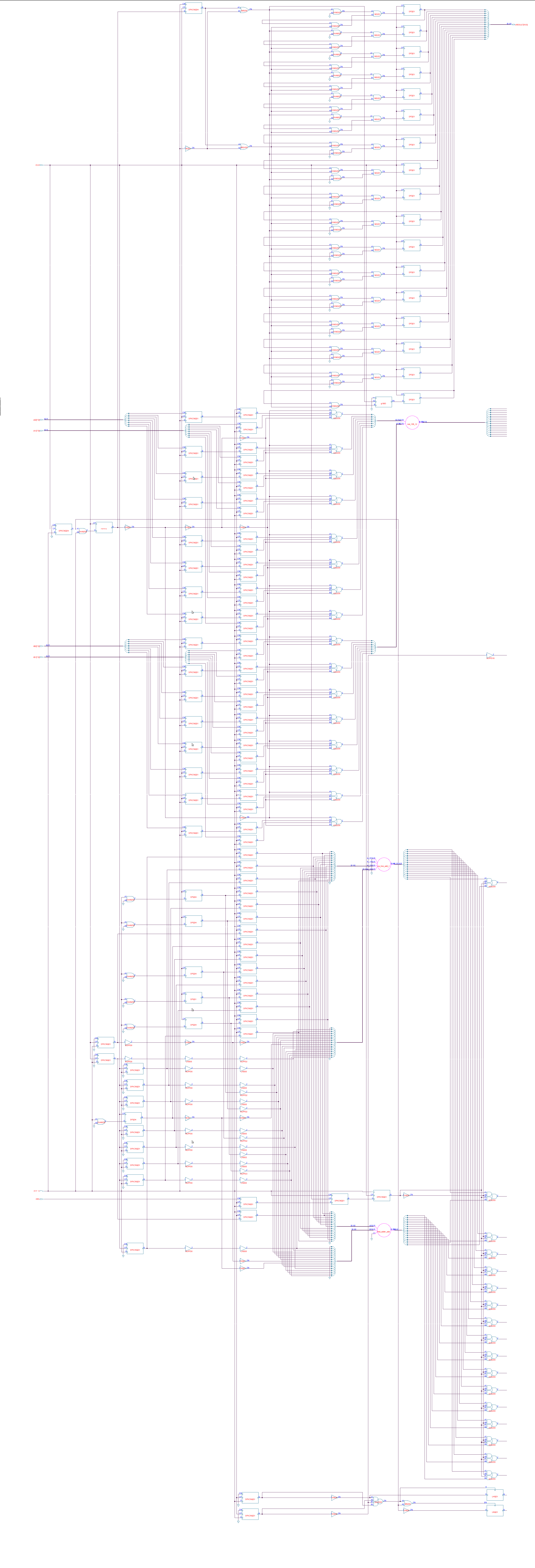
report power **>** **${**OUTPUT\_DIR**}/**power.rpt

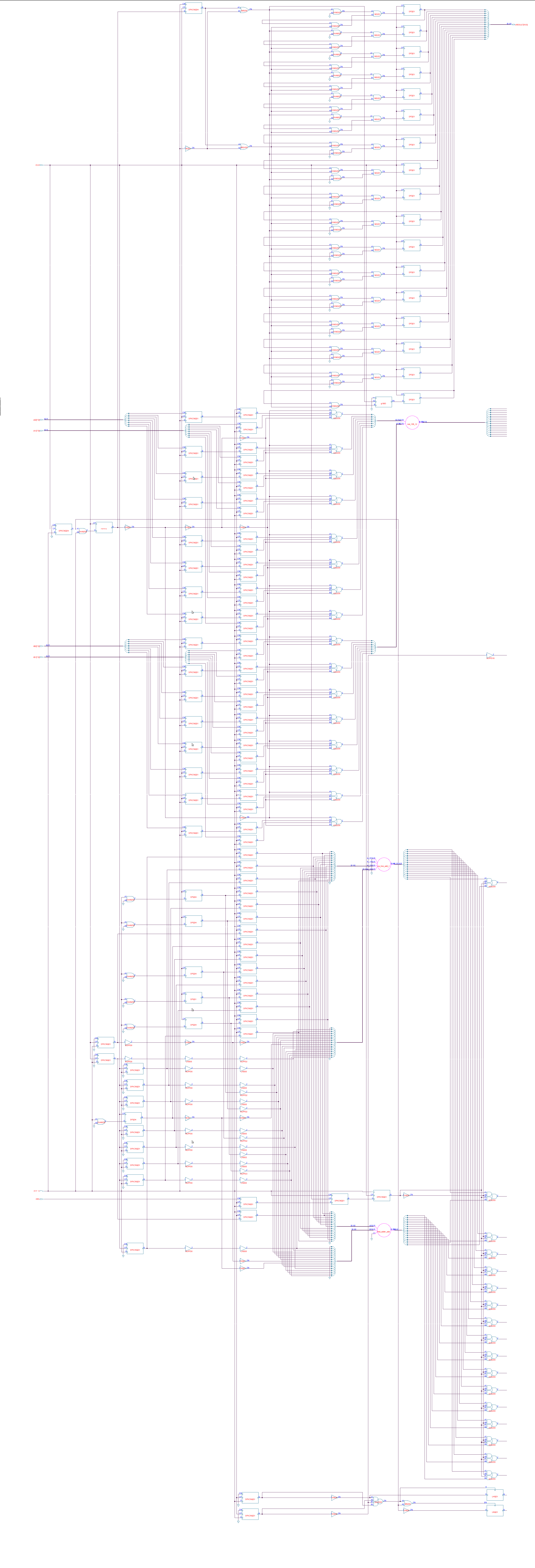
write -mapped **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.v

write\_script **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.g

write\_sdc **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.sdc

This is the synthesis with no clock gating enabled. Which resulted in the synthesized circuit on the next page.





## Power, Area and Timing reports:

### Power:

Leakage Dynamic Total

Instance Cells Power(nW) Power(nW) Power(nW)

--------------------------------------------------------------------

ALT\_MULTADD\_9 788 25576.140 3670617.854 3696193.994

csa\_tree\_a..116\_30\_groupi 392 10639.582 0.000 10639.582

mul\_106\_12 135 5552.124 0.000 5552.124

final\_adder\_add\_113\_21 18 1633.508 0.000 1633.508

### Area:

### Instance Cells Cell Area Net Area Wireload

### ---------------------------------------------------------------------------

### ALT\_MULTADD\_9 788 3167 0 ZeroWireload (S)

### csa\_tree\_add\_116\_30\_groupi 392 1377 0 ZeroWireload (S)

### mul\_106\_12 135 590 0 ZeroWireload (S)

final\_adder\_add\_113\_21 18 144 0 ZeroWireload (S)

### Timing:

Pin Type Fanout Load Slew Delay Arrival

(fF) (ps) (ps) (ps)

---------------------------------------------------------------

(clock clk) launch 10 R

EN\_3\_reg/CP 0 10 R

EN\_3\_reg/Q DFKCNQD4 2 13.6 31 +123 133 F

g1946/A1 +0 133

g1946/ZN NR2XD8 17 19.4 35 +29 162 R

g1669/A2 +0 162

g1669/ZN CKND2D1 1 1.2 25 +28 190 F

g1668/A1 +0 190

g1668/ZN ND2D1 1 1.1 21 +20 210 R

oRESULT\_reg[9]/D DFQD1 +0 210

oRESULT\_reg[9]/CP setup 0 +30 240 R

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

(clock clk) capture 210 R

---------------------------------------------------------------

Timing slack : -30ps (TIMING VIOLATION)

Start-point : EN\_3\_reg/CP

End-point : oRESULT\_reg[9]/D

## Clock gating report:

With the command “report clock\_gating -summary” I got the following report:

## Summary

## -------

## ---------------------------------------------

## Category Number %

## ---------------------------------------------

## RC Clock Gating Instances 0 -

## Non-RC Clock Gating Instances 0 -

## ---------------------------------------------

## RC Gated Flip-flops 0 0

## Non-RC Gated Flip-flops 0 0

## Ungated Flip-flops 105 100

## Total Flip-flops 105 100

---------------------------------------------

Obviously because I didn’t enable clock gating there were no clocks gated.

## Synthesis with clock gating enabled:

## This sets the name of the directory in which area/timing/power reports

## and synthesized (mapped) netlists are stored.

**set** OUTPUT\_DIR **./**run\_dir

**if** **{** **![file** exists **${**OUTPUT\_DIR**}]** **}** **{** sh mkdir **${**OUTPUT\_DIR**}** **}**

#### Step 1 ####

## This tells the compiler where to look for the libraries

set\_attribute lib\_search\_path **../**libdir

## This defines the libraries to use

set\_attribute library **{**tcbn65gpluswc.lib**}**

##set\_attribute library {tcbn65gplustc.lib}

##set\_attribute library {tcbn90ghpbc\_ccs.lib}

##set\_attribute lp\_insert\_clock\_gating true

#set\_attribute lp\_insert\_operand\_isolation true

#rm /designs/\*

#### Step 2 ####

##This must point to your VHDL/verilog file

**load** -v2001 **../../**ALT\_MULTADD\_redid.v

set\_attribute lp\_insert\_clock\_gating true

#### Step 3 ####

## This builds the general block

elaborate

dc::set\_time\_unit -nanoseconds

dc::set\_load\_unit -picofarads

define\_clock -period 200 -name clk **[**dc::get\_ports **{**iCLK**}]** -rise 5 -fall 5

set\_attribute lp\_power\_unit **{**nW**}**

set\_attribute max\_leakage\_power 100000 **/**designs**/**ALT\_MULTADD\_9

set\_attribute power\_optimization\_effort high

synthesize -to\_mapped -effort high

report area **>** **${**OUTPUT\_DIR**}/**area.rpt

report gates **>** **${**OUTPUT\_DIR**}/**gates.rpt

report timing **>** **${**OUTPUT\_DIR**}/**timing.rpt

report timing -lint **>** **${**OUTPUT\_DIR**}/**lint.rpt

report summary **>** **${**OUTPUT\_DIR**}/**summary.rpt

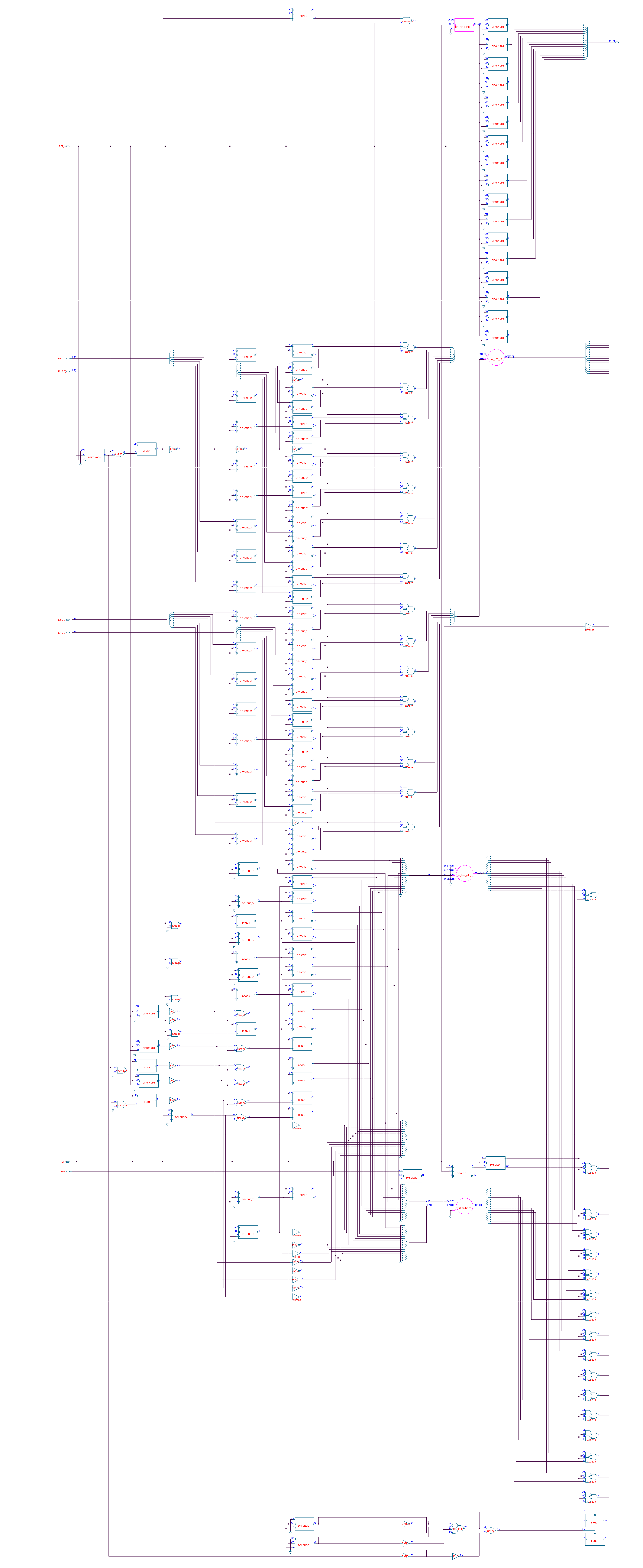
report power **>** **${**OUTPUT\_DIR**}/**power.rpt

write -mapped **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.v

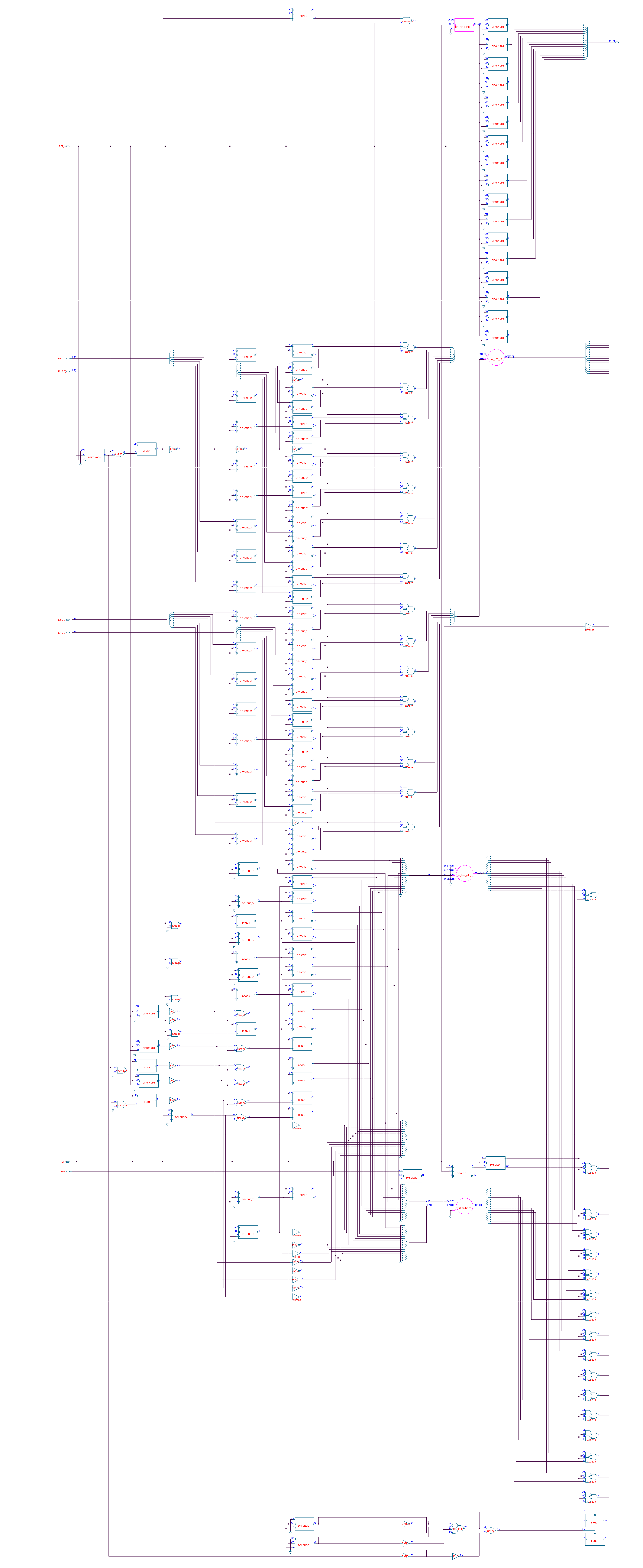
write\_script **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.g

write\_sdc **>** **${**OUTPUT\_DIR**}/**jpeg\_mapped.sdc

Clock gater



Shared multiplier



Control logic

## Power, Area and Timing reports:

### Power:

#### Leakage Dynamic Total

#### Instance Cells Power(nW) Power(nW) Power(nW)

#### --------------------------------------------------------------------

#### ALT\_MULTADD\_9 714 24911.327 3187026.832 3211938.160

#### csa\_tree\_a..116\_30\_groupi 379 10296.895 0.000 10296.895

#### mul\_106\_12 135 5552.124 0.000 5552.124

#### final\_adder\_add\_113\_21 18 1633.508 0.000 1633.508

#### RC\_CG\_HIER\_INST0 1 41.593 23555.208 23596.801

Dynamic and total power consumption decreased as expected due to the clock gating being enabled.

### Area:

Instance Cells Cell Area Net Area Wireload

---------------------------------------------------------------------------

ALT\_MULTADD\_9 714 3080 0 ZeroWireload (S)

csa\_tree\_add\_116\_30\_groupi 379 1356 0 ZeroWireload (S)

mul\_106\_12 135 590 0 ZeroWireload (S)

final\_adder\_add\_113\_21 18 144 0 ZeroWireload (S)

RC\_CG\_HIER\_INST0 1 6 0 ZeroWireload (S)

Area decreased due to replacement of AOI gates with the clock gate.

### Timing:

Pin Type Fanout Load Slew Delay Arrival

(fF) (ps) (ps) (ps)

--------------------------------------------------------------

(clock clk) launch 10 R

EN\_3\_reg/CP 0 10 R

EN\_3\_reg/QN DFKCND4 1 1.9 16 +154 164 R

g1263/A1 +0 164

g1263/ZN CKND2D2 1 1.0 14 +15 178 F

RC\_CG\_HIER\_INST0/enable

RC\_CGIC\_INST/E CKLNQD1 +0 178

RC\_CGIC\_INST/CP setup 0 +47 226 R

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

(clock clk) capture 210 R

--------------------------------------------------------------

Cost Group : 'cg\_enable\_group\_clk' (path\_group 'cg\_enable\_group\_clk')

Timing slack : -16ps (TIMING VIOLATION)

Start-point : EN\_3\_reg/CP

End-point : RC\_CG\_HIER\_INST0/RC\_CGIC\_INST/E

Timing slack improved slightly.

## Clock gating report:

With the command “report clock\_gating -summary” I got the following report:

Summary

-------

---------------------------------------------

Category Number %

---------------------------------------------

RC Clock Gating Instances 1 -

Non-RC Clock Gating Instances 0 -

---------------------------------------------

RC Gated Flip-flops 17 16

Non-RC Gated Flip-flops 0 0

Ungated Flip-flops 88 84

Total Flip-flops 105 100

---------------------------------------------

# Conclusion

As we have mentioned before, the coding style dramatically effects the synthesized circuit. We can see this clearly now that we have changed the code to be more modular and to not set registers in multiple always blocks. There are improvements in all specifications of the circuit. As for the changes caused by the clock gating, it is hard to tell if it alone improved specifications because it may be only due to the changed coding style. I had a hard time getting the clock gating to synthesize correctly but finally found that I needed to put the if statement in the always block for the registers which I was clock gating.